**AWS Project Build a Game with a Continuous Deployment Pipeline from GitHub to S3**

Creating a continuous deployment pipeline for your memory matching game using AWS Code Pipeline and S3 involves several steps. This guide will walk you through the process, from setting up your S3 bucket for static website hosting to creating the deployment pipeline.

### Step 1: Set Up Your S3 Bucket for Static Website Hosting

1. Log in to the AWS Management Console and navigate to the S3 service.

2. Create a new S3 bucket. Choose a unique name for your bucket.

3. Enable static website hosting\*\* for your bucket. Under the "Properties" tab, find the "Static website hosting" section and click "Edit". Select "Enable" and specify the index document (usually `index.html`).

4. Set bucket policy to make your website publicly accessible. Go to the "Permissions" tab, click "Bucket Policy", and add a policy that grants public read access to your bucket.

\*\*\* bucket policy file is in my repository\*\*\*

A screenshot of a computer

Description automatically generated

Write bucket name

* Write unique S3 bucket name.
* Select create bucket.

A screenshot of a computer

Description automatically generated

* Bucket is created.
* Click on the bucket name.
* Go to permission tab.

A screenshot of a computer

Description automatically generated

* Select Bucket Policy.
* Click on edit.

A screenshot of a computer

Description automatically generated

* Write the bucket policy.
* This bucket policy sets get an object.
* Write your bucket ARN followed by ‘ /\* ‘.

A white screen with text

Description automatically generated

Write you bucket ARN here.

* Click on save changes.
* Select the properties tab and scroll to bottom and select Static Web hosting.

A screenshot of a computer

Description automatically generated

* Enable it .
* In index document tab write index.html.
* Click on save changes.
* After that you will get any website URL, which will be able to display the content of the website.

A screenshot of a web page

Description automatically generated

A white rectangular object with a black border

Description automatically generated

### Step 2: Prepare Your Game Code

Ensure your game code (HTML, CSS, and JavaScript) is ready and organized in a GitHub repository. Your repository should have a structure that reflects the static website hosting setup, with `index.html` at the root.

\*\*\* the code for the website is on my GitHub repository and the images used. All the content used for the website is on my GitHub repository.

### Step 3: Create an AWS CodePipeline

1. Navigate to the AWS CodePipeline service in the AWS Management Console.

2. Create a new pipeline. Name it appropriately, e.g., "MemoryMatchGameDeployment".

3. Choose the source provider. Select GitHub as your source provider and connect your GitHub account. Choose the repository and branch where your game code is hosted.

4. Choose the build provider. Since your game consists of static files (HTML, CSS, JavaScript), you don't need a build step. You can skip this step.

5. Choose the deploy provider. Select Amazon S3 as your deploy provider. Configure it to deploy to the S3 bucket you created earlier.

6. Review and create the pipeline. AWS CodePipeline will automatically detect changes in your GitHub repository and deploy them to your S3 bucket.

* Go to Codepipeline.
* Create Codepipeline.
* Write the name of the pipeline.
* Pipeline type P1.
* Select the new role, it will automatically create a new role.

A screenshot of a computer

Description automatically generated

* Select next.
* Select the source provider, here we are using GitHub so select GitHub version2.
* When you click on next a window will pop up to get the connection from GitHub.
* Write the connection name.
* Click on connect to GitHub.

A screenshot of a computer

Description automatically generated

* Connect to your github account.
* Authorize it.
* Click on only repository and select the uploaded code repository.

A screenshot of a computer

Description automatically generated

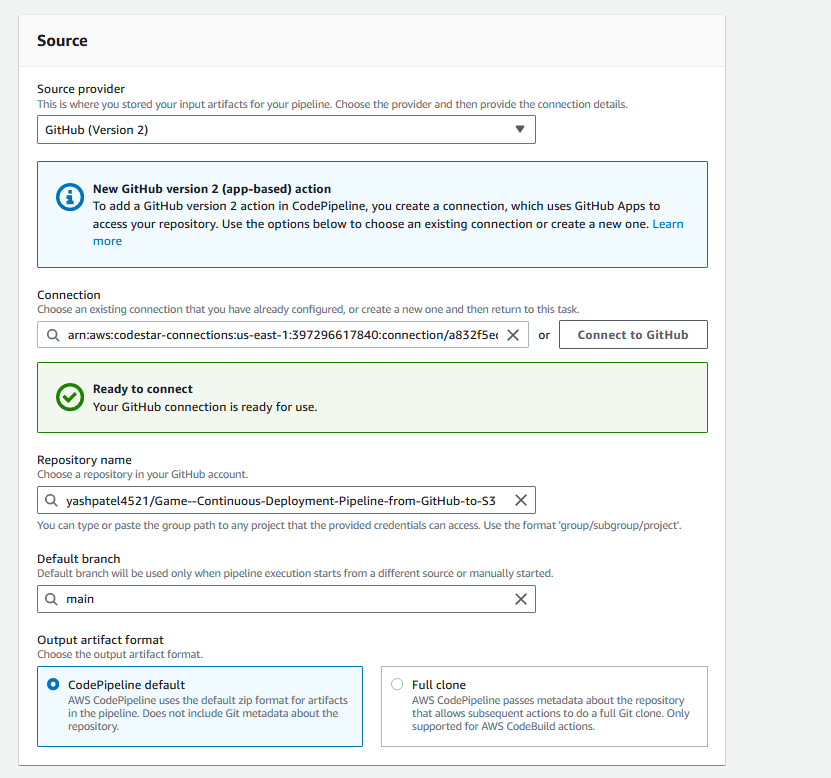
A screenshot of a computer

Description automatically generated

* Click on install.
* The app will then install.
* You can se some numbers generated.
* Then click on connect.

A screenshot of a computer

Description automatically generated



* Select the connection you created.
* Select the repository name.
* Default branch as main.
* Click next.
* Skip the built stage.
* Add Deploy, so we are deploying on S3 bucket so select s3 bucket.
* Select the bucket which you created.
* Tick the Extract file before deploy.
* Click next.

A screenshot of a computer

Description automatically generated

* The deployment stage has begin deploying the resources into S3 bucket.

A white and grey background with a black and white stripe

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

* Finally its been deployed !!! .
* Go to the static website URL and copy it to the browser and see its up and running the website.
* When you click on the start button you can now see that the game is running.

A white rectangular object with a black border

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a game

Description automatically generated

### Step 4: Test Your Deployment

1. Make a change to your game code in your GitHub repository.

2. Commit and push the change to the branch you're using with AWS CodePipeline.

3. Monitor the pipeline execution in the AWS CodePipeline console. You should see the pipeline automatically triggering a deployment upon detecting the change.

4. Access your game by navigating to the S3 bucket's static website endpoint (found in the bucket properties under "Static website hosting").

* Now we will make some changes to the index.html file and let’s see how automatically it deploys on the S3 bucket.
* Makin the changes to the line no 13. (Like adding some \*\*\*\* welcome to the meme matching game \*\*\*\*\* ) something like that.
* Then commit the changes. It will automatically trigger the code pipeline to make the changes to the s3 bucket where it is deployed.
* No refresh the web URL and see the changes have been done.

A screenshot of a computer

Description automatically generated

A white rectangular object with black lines

Description automatically generated with medium confidence

A screen shot of a computer

Description automatically generated

Successfully !!!!!

### Conclusion

By following these steps, you've set up a continuous deployment pipeline for your memory matching game. This setup allows you to easily update your game by pushing changes to your GitHub repository, and AWS CodePipeline will automatically deploy these changes to your S3 bucket. You can now focus on enhancing your game with additional features and improving the user experience.